DUPLICATE ALSO

Met O (APR) Turbulence and Diffusion Note No. 249

The parallelisation of the 3D Lagrangian global Chemistry transport model (STOCHEM)

by

W.J.Collins, and C.E.Johnson

June 1998

Met O (PMSR) (Public Met. Services Research)
Meteorological Office
London Road
Bracknell
Berkshire, RG12 2SZ

This work was supported through the Public Meteorological Service research and development programme of the Meteorological Office as part of the research programme of the Air and Environment Quality Division of the Department of the Environment, Transport and Regions through contract number EPG 1/3/93, and as part of the Climate Prediction programme of the Global Atmosphere Division of the Department of the Environment, Transport and Regions through contract number PECD 7/12/37.

Note:

This paper has not been published. Permission to quote from it should be obtained from the Head of Public Met. Services Research Branch, MetO(PMSR).

© Crown copyright 1998

ORGS UKMO T

National Meteorological Library
FitzRoy Road, Exeter, Devon. EX1 3PB

NATIONAL
METEOROLOGICAL
LIBRARY
Bracknell RG12 2SZ

The parallelisation of the 3D Lagrangian global chemistry transport model (STOCHEM)

W.J.Collins C.E.Johnson

June 1998

1 Introduction

In February 1998 the vector processing computer, the CRAY C90, was decomissioned. To prepare for this the STOCHEM chemistry model (Collins *et al.* 1997; Stevenson *et al.* 1997) had to be redesigned to run on the replacement machine, the CRAY T3E massively parallel computer. STOCHEM is an offline chemistry transport model that reads in 6 hourly meteorological data either from the operational archive, or from climate simulations. These data are used to control the advection and chemistry of around 50,000 cells distributed around the globe, which hold the concentration of 70 chemical species. In principle the FORTRAN code would still have run on the new machine on one processing element (PE). However each PE on the T3E runs about a factor of 5 slower than a CPU on the C90, which would have been an unacceptable decrease in performance. Also, most of the PEs have only 128Mbytes of memory which would have been insufficient to hold all the meteorological fields and species concentrations. Most importantly, the switch to the new machine provided an opportunity to dramatically speed up the running of the chemistry model by parallelising the code so that it runs on more than one processor.

Routine	CPU time (s)	percentage of total
Chemistry	2108	83
Photolysis	217	9
Advection	116	5
Reading files	1	0
Others	91	3
Total	2533	100

Table 1: CPU time and percentage CPU taken by the most CPU intensive routines for a 1 day model run on one T3E processor. These are not the same percentages as when the model was run on the CRAY C90, since for that machine the chemistry was optimised for vector processing.

The two advantages of running a code on several processors are that the processing time and data storage can be split between the processors, such that each processor needs to compute a fraction of the calculations and hold a fraction of the data. In STOCHEM, the main computation loads were in the integration of the chemical reactions, with calculation of the photolysis rates and advection of the Lagrangian cells also important (see table 1). The advection and the chemistry are decoupled, using a 3 hour timestep for the advection, and a 5 minute timestep for the chemistry. Photolysis rates are calculated every 45 minutes. The largest amounts of

data needed were the arrays containing the meteorological data even at the climate resolution of the UM (see table 2). Therefore the priority was to parallelise the chemistry as efficiently as possible with optimum load balancing between processors, and to parallelise the photolysis and advection with the loads as well balanced as was practical. The meteorological arrays needed to be split between processors so they no longer dominated the data storage. This will become even more important when we wish to add more meteorological fields, or use the higher resolution old UM operational fields or even the very high resolution new operational fields. At the same time as parallelising the code, we decided to convert the code from FORTRAN 77 to Fortran 90 in order to make use of its new features, especially the array processing facilities.

Arrays	Number of elements	Percentage of total
Meteorological fields	3945K	30
Cell information	3803K	28
Eulerian concentrations and fluxes	3033K	23
Photolysis rates	1586K	12
Others	944K	7
Total	13311K	100

Table 2: Number of elements in each array and percentage of total. The size of the meteorological fields is for the climate resolution. For the operational resolution these numbers should be multiplied by 5.

2 Division by region

It is possible to run on up to 180 PEs, with a requirement for the job scheduling that the number of PEs used must be an integer multiple of 12. Initially we have chosen to run on 36 processors as this seems a sensible compromise between spreading the load as widely as possible and reducing the amount of data transfer between PEs. It will be useful to test whether increasing the number of PE used reduces the run time without too large a penalty in the rate of job scheduling.

The most obvious way to divide the work between a set of PEs is for each PE to process a different geographical region. Certain operations, such as mixing between cells, would become very difficult if nearby cells were distributed among many different processors. The surface of the globe is split into equal latitude bands. These bands are then split longitudinally into varying numbers of columns so that each region covers approximately the same surface area (the latitude bands containing the poles are not split). Each processor is then responsible for one region, extending the full height of the model (see figure 1 for an example using 36 PEs).

In the model the definition of nearby is that the cells occupy the same Eulerian $5° \times 5°$ grid square. This is an unphysical definition since cells at the edge of a grid square may be nearer to some cells in neighbouring grid squares than to those in their own grid square. If cells were allowed to interact with those in other grid squares this would cause problems at the edges of regions where the cells would be on different processors.

The meteorological data are split up into the same regions, but with overlapping latitude bands. The overlaps are large enough (13.75°) so that cells are never advected to a point beyond their originating latitude band in one timestep. There is no division in the longitude direction since near the poles cells can be advected with a very large change in longitude. Regions further

from the poles could be split into longitude ranges, but it was thought to be easier to program if all the meteorological arrays had the same latitude and longitude dimensions. This means that several processors hold the same meteorological subarrays.

The other geographically based arrays (Eulerian concentrations/fluxes, photolysis frequencies, emissions rates and deposition velocities) could also be split up between processors. However reducing the size of the meteorological arrays was all that was necessary to fit the model onto the T3E processors. While it would have been more elegant to reduce the memory requirements still further, it was not a priority. Therefore in the initial version of the model (version 0) all the processors hold Eulerian data arrays covering the globe. This means that on any processor a large fraction of the elements of each array (those corresponding to regions not covered by the processor) are undefined. Each processor runs the photolysis code only for the region which it covers, and similarly accumulates Eulerian concentrations and fluxes only for this region. To write out Eulerian arrays, for output data and program dumps, the arrays are first collected together on one processor. This can result in the transfer of a lot of data between processors, so it is important that this is only done when necessary. Data output is usually only done once a month, but program dumping once a day proved too time consuming and it was decided to do this once every ten days. This means however that there is less flexibility when restarting the model from dump files.

Cells are allowed to move freely between all the regions. If cells move out of the geographical region for a processor after advection, they are swapped to their new processor, passing over all the information on cell position and species concentrations.

3 Balancing the computational load

The geographical division of cells gives roughly, but not exactly, equal numbers of cells on each processor. Since over 80% of the computational load used to be in the chemistry integration, it was thought essential to balance the load between processors as evenly as possible by pooling the cells and redividing them. This is done by swapping extra cells on processors responsible for more than the average number of cells onto processors responsible for less than the average, so that all processors are responsible for the same number of cells (± 1). Unfortunately this means that the link between processors and geographical region is now broken, but the chemistry is now computed more efficiently.

All the information needed to calculate the emission, deposition and photolysis is based on geographical data arrays. Therefore it will not be available on a processor that has had cells swapped in from outside its geographical region in order to balance the load. It would be simplest to calculate this information before pooling the cells. Unfortunately, with the present program structure, this would mean storing arrays of $\sim$ 3500K elements each, bringing the total storage required above the memory limit for each processor. Instead each time a cell is swapped from one processor to another, the Eulerian data for the corresponding $5^\circ \times 5^\circ$ grid square is copied over too. For 3 dimensional arrays, the data for the whole vertical column is copied.

After the chemical integration, the cells have to be swapped back to their original processors (geographically allocated). The remaining routines (convective redistribution, interparcel mixing, and collection of statistics) need the cells to be grouped geographically and are not very computationally expensive. This conflict between routines that need nearby cells to be located on the same processor, and routines than require cells to be as evenly divided between processors as possible, means that cells frequently need to be moved between processors.

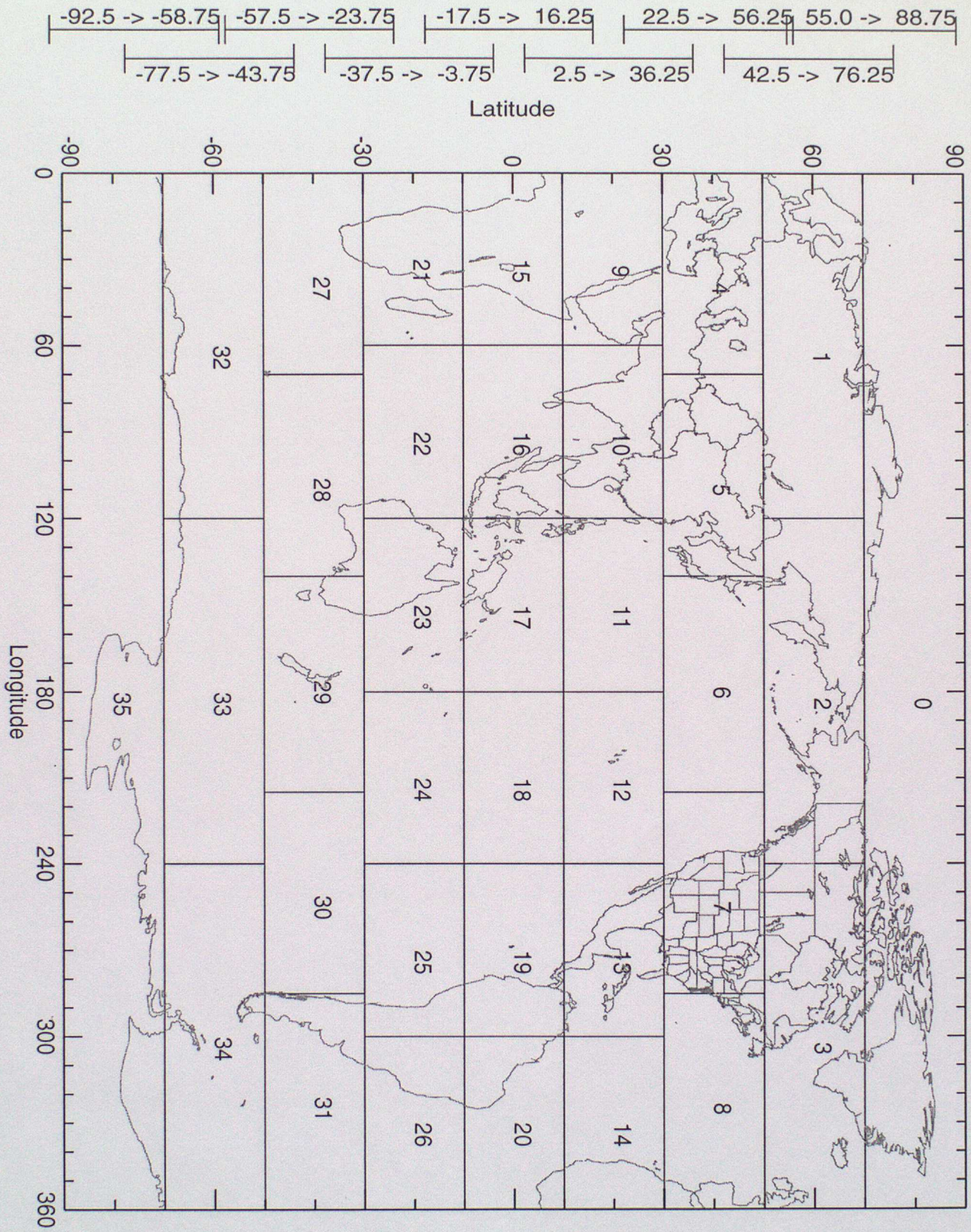

Figure 1: The PEs associated with each geographical region with for 9 rows and (1,3,5,6,6,6,5,3,1) columns. The ranges on the left hand side show the latitudinal extents of the meteorological data fields.

4 Other changes to the model

The array containing the Eulerian chemical concentrations at each timestep contains zeroes where there are no cells occupying a particular grid volume. This occurs mostly in the high latitudes, but also occasionally even in the tropics. This was overcome in the original model by smoothing the Eulerian concentrations in 2 dimensions (longitude-latitude) with a Gaussian weighting function. This is not so easy with parallel processing since different regions of the array are defined on different processors. The zeroes in the arrays cause problems in accumulating concentration statistics, and for the tropospheric ozone profile input to the photolysis calculations. The first problem was solved by keeping track of the number of times a non-zero concentration had been added to each grid volume. The ozone profile problem was solved by keeping a running average of the ozone concentration, using the following algorithm to calculate the new values after each timestep:

$$\begin{aligned} &\text{if } C_{ijk} = 0 \quad \text{then} \quad O3_{ijk}^{t+\Delta t} = O3_{ijk}^{t} \\ &\text{if } C_{ijk} \neq 0 \quad \text{then} \quad O3_{ijk}^{t+\Delta t} = (1-\epsilon)O3_{ijk}^{t} + \epsilon C_{ijk}, \end{aligned}$$

where C_{ijk} is the instantaneous ozone concentration for grid volume (i, j, k), $O3_{ijk}$ the running average ozone concentration, t is the time at the previous timestep, Δt is the model advection timestep, and ϵ is a constant equal to $\Delta t/(10 \text{ days})$.

Some changes to the model unconnected with parallelisation were the addition of snow and ice cover fields for use in calculating the surface deposition velocities and surface albedoes.

5 Timing and performance optimisation.

Additional run time is involved in transferring data between processors. We have used two routines GC_RSUM and GCG_RALLTOALLE to do this, taken from the GCOM library (Amundsen and Skaølin 1996). GC_RSUM calculates the sum of a variable or array across all the processors and distributes the result to all the processors. GCG_RALLTOALLE is much more flexible and can send different subsections of an array to different processors. During the installation of the T3E the number of array elements that could be sent between processors by these two routines was increased from 2048 to 65536. We found that we could save substantial run time by taking advantage of this increase through sending 3 dimensional arrays as single entities rather than level by level.

A problem when timing routines in the parallel processing environment of the T3E is that the CPU time taken by a routine on each processor can include idle time waiting for other processors to have reached the same point in the calculation. Each time the program needs to send data between processors it synchronises them so they can all continue from the same point. If the computational load between processors is not balanced, this waiting time can become significant as some processors finish long after others. There is little time lost if one processor arrives at a synchronisation point much earlier than the rest. However if one processor arrives much later than the rest, then all the others (35 processors in our case) have to waste valuable processing time waiting for it.

We first timed the routines in the parallelised code for a 10 day run starting from hard-wired initial conditions, for which the average CPU time per PE was 1424 seconds. It appeared that a significant proportion of the time (22% on some PEs) was being taken up in the pooling of

the cells to balance the load for the chemistry, even though the amount of computation done in the pooling was small. This seemed to be confirmed when we removed the pooling routines, since the program then ran about 4% faster (50s faster for a 10 day run). Although we were pleased at any increase in speed, in a way this was quite disheartening as a considerable amount of effort had been put in to develop the load balancing and make it work.

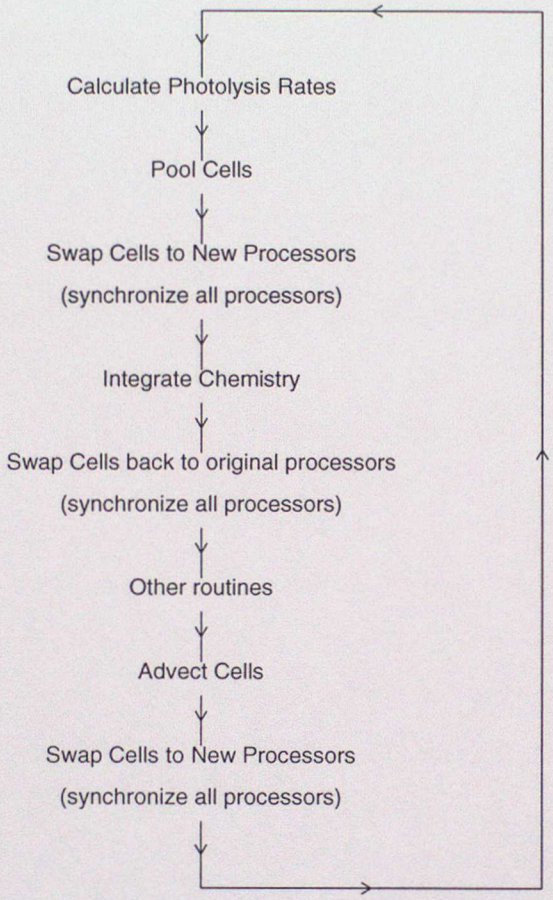

Figure 2: Schematic flow diagram of main program loop

A schematic flow diagram of the main program loop is shown in figure 2. When pooling was taken out, so too were the first two swapping routines and their synchronisation calls. This meant that there was no processor synchronisation from before the photolysis calculations until after the advection calculation. The conclusion was that the time saved by not having to wait for the synchronisation more than made up for the time lost by not load balancing the chemistry. The long wait for the synchronisation turned out to be mostly due to the difference in time taken for the photolysis calculation (250 s). A smaller time penalty was incurred after the chemical integration, since even though the same number of cells were processed on each PE, the time taken in the calculation varied by up to 60 s.

Even though the two processors representing the poles cover the same surface area as the other processors, they have to process the photolysis rates for four times more $5^\circ \times 5^\circ$ grid squares than the average. It is difficult to balance the load for the photolysis calculation exactly since not only do different processors cover different numbers of grid squares, but the fraction of these that are in light or darkness varies too. In theory some sort of pooling and redistribution could be done to give each processor the same number of daylight grid squares to process. However it is not clear that the time saved would make up for the extra computation needed. Given that the greatest benefit can be gained by speeding up the few slowest processors, a simple fix to the problem was to only calculate the photolysis rates for every sixth grid square in the longitudinal

direction for the polar regions. Intermediate points were filled in by linear interpolation. This does degrade the longitudinal resolution of the polar photolysis rates, but only brings them into line with the rest of the globe since divisions in longitude are much more closely spaced near the poles compared with midlatitudes or the tropics. This change saved an extra 160 s for a 10 day run, bringing the CPU time down to 1215 s per processor.

Having made this change it still did not seem profitable to balance the loads for the chemistry integration. In fact load balancing now added an extra 100 s to the run. This is still due to the timing differences in the both the photolysis calculation and the chemistry. Although the photolysis calculation has been speeded up at the poles, there are still differences between the PE since at each timestep, some are processing regions entirely in daylight, some regions entirely in the dark, and some partially lit. This causes most problems in the latitude bands one away from the poles where the regions cover twice as many grid squares as the tropical ones.

The cells are initialised evenly with surface area in the model, so after 10 days the maximum number of cells processed by one PE was 1498, compared with a mean of 1390 (an 8% difference). This does not offer scope for large savings by load balancing, so it is not surprising that the load balancing was inefficient. However for longer runs the story is different, since for some reason that is not currently apparent, cells tend to accumulate in certain locations, in particular around the South Hemisphere Jet Stream. To examine the effect of this, the model was run for 10 days again, but this time starting from a model dump after a 10 month run. The maximum number of cells processed by one PE was then 1906 (37% greater than the average). Without load balancing the CPU time increased to 1465 s per processor for the 10 day run. Load balancing brought this down to 1255 s, a saving of 210 s or 17%. A timing profile is shown in figure 3. Assuming these 10 days were typical, the CPU time required for each processor for a year's model run will be slightly under 13 hours which is about 7 times faster than the 90 hours per year the previous model took on the CRAY C90. Scaling up the timings from the non-parallel run in table 1 would give 257 hours for a year, a factor of 20 longer than using 36 processors. That this factor is less than 36 shows that there are considerable inefficiencies associated with parallelising the code. When job scheduling time is taken into account, the model will run for a year in about 24 hours, which compares very favourably with other models such as the NCAR model MOZART which with similarly detailed chemistry takes over 100 CPU hours per year (Brasseur *et al.* 1996).

Routine	CPU time (s)	percentage of total
Chemistry	720	57
Photolysis	37	3
Advection	47	4
Reading files	135	11
Pooling and swapping	224	18
Others	92	7
Total	1255	100

Table 3: CPU time and percentage CPU taken by the most CPU intensive routines for a typical processor in a 36 processor 10 day model run.

6 Improvements

6.1 Cell variables

After the meteorological data, the next largest data arrays are those containing the cell concentrations and positions. These are large enough to hold information on all the 50048 cells. Most of this space is redundant since each processor is responsible for only a subset of all the cells. This waste of space would prevent us from increasing the number of cells substantially (say to 100K or 200K).

For the next model version (version 1) the arrays holding cell information now have a dimension which is set to the average number of cells per processor plus 67%. The extra length was found necessary to allow for an uneven number of cells on each processor and as storage when swapping cells from one processor to another. Previously, when swapping elements of cell arrays the elements were simply copied from one processor to the same position in another processor. Now elements are taken from the middle of the array on one processor, leaving a hole, and added to the end of the array on the destination processor. After all elements have been swapped, the holes in each array are filled by taking elements from the end of the array. Hence cells will no longer have an invariant cell number, but will change when swapped. This will add a complication when trying to follow individual cells, which is a facility that we do not use at present, but might like to in the future.

The treatment of all Eulerian data is the same as in version 0. The time penalty in version 1 compared with version 0, due to the extra computation involved in swapping cells is negligible.

6.2 Eulerian data

As mentioned in section 2 it was decided not to split the Eulerian data arrays into regions for version 0 of the model. For consistency with the meteorological data and to remove the last class of redundant array storage, it makes sense to create a new model version (version 2) where the dimensions of the Eulerian data arrays are reduced to the sizes of the geographical regions covered by each processor. This will allow us to store the 3 dimensional concentrations of more of the chemical species and reaction fluxes, or to double the number of vertical levels in the output.

As with the meteorological data, it is simplest if the arrays all have the same latitudinal and longitudinal dimensions, so they are split into 9 (for a 36 processor run) latitude bands but not split in the longitude direction. Splitting the arrays causes a slight complication when outputting data fields as the data cannot be collected together on one processor. Instead all the data for each latitude band have to be collected onto just one of the processors (the first) responsible for that band. Data are then written from each of these 9 processors in turn, instead of just from one. Our normal format for the output fields is to write out the full longitude-latitude grid one level at a time and then loop over each chemical species or reaction flux. This becomes very inefficient since the processors have to be synchronised before each latitude band is written out. A more efficient way would be to write out from the first processor the first latitude band for all the levels and all species/fluxes, and then the next latitude band from the next processor and so on.

The other complication is that, as before, when the cells are reallocated for optimum load

balancing they are no longer allocated on the basis of geographical location. Previously this was overcome by copying data for individual grid squares or columns between processors. This is not possible any more as the Eulerian data arrays no longer cover the whole globe. However, since the size of the arrays containing cell information have been reduced (in version 1), emissions, dry and wet depositions, and photolysis rates can now all be calculated before reallocating the cells, and stored as cell information. Now all the data can be swapped on a cell basis which is much neater than the previous method of swapping half on a cell basis and half on a Eulerian grid basis.

7 Summary and status of the different versions

Version 0 has the processing split between 36 PEs, meteorological grids split into 9 latitude bands according to PE, and each PE holding full sized cell and Eulerian data arrays with only a subset of the elements defined. Version 1 is as version 0, but each PE holds smaller cell arrays big enough to store information on all the cells processed by that PE. Version 2 is as version 1, but the Eulerian data arrays are split into 9 latitude bands.

Versions 0 and 1 have been tested as fully as is practicable and have been used with climate resolution meteorology. They give virtually identical results to each other and to the model that was run on the CRAY C90. They should be easy to convert to run on the old operational grid.

Version 2 is in the process of being tested. At present the results are very similar, but not identical, to the other versions.

Acknowledgements

This work was supported through the Public Meteorological Service research and development programme of the Meteorological Office, as part of the research programme of the Air and Environment Quality Division of the Department of the Environment, Transport and Regions through contract number EPG 1/3/93 and as part of the Climate Prediction programme of the Global Atmosphere Division of the Department of the Environment, Transport and Regions through contract number PECD 7/12/37.

References

Amundsen, J. and Skøalin, R., 1996, *GC User's Guide, Release 1.0.4*, SINTEF Applied Mathematics, Norway.

Brasseur, G.P., Hauglustaine, D.A. and Walters, S., 1996, Chemical compounds in the remote Pacific troposphere: Comparison between MLOPEX measurements and chemical transport calculations. *J. Geophys. Res.*, **101**, 14795–14813.

Collins, W.J., Stevenson, D.S., Johnson, C.E. and Derwent, R.G., 1997, Tropospheric ozone in a global-scale three-dimension Lagrangian model and its response to NO_X emission controls. *J. Atmos. Chem.*, **26**, 223–274.

Stevenson, D.S., Johnson, C.E., Collins, W.J. and Derwent, R.G., 1997, Changes to tropospheric oxidants from aircraft nitrogen oxide emissions studied with a 3-D Lagrangian model. *Atmos. Environ.*, **31**, 1837–1850.